Scheduler Specification

Embedded Systems

Francisco Martinez Chavez

2-Sep-18

ITESO

Universidad Jesuita
de Guadalajara

First version by	Abraham Tézmol Otero
Specification updated by	Francisco Martínez Chávez
Date	2-Sep-18

1. TABLE OF CONTENTS

2. FUNCTIONAL SPECIFICATION

This software module provides 4 non-preemptive execution threads as follows:

- 1ms thread
- 10ms thread
- 50ms thread
- 100ms thread

In order to balance CPU workload, this module implements two independent execution threads as follows:

- 2ms A thread
- 2ms B thread

Such threads share execution time with the previously defined execution threads as per the following arrangement:

PRIMARY THREAD	*DEPENDENT THREAD*
1MS THREAD	100MS THREAD
2MS A THREAD	50MS THREAD
2MS B THREAD	10MS THREAD

Due to the non-preemptive nature of the task scheduler and the periodicity of the fastest execution thread (1ms), each combination of primary thread and dependent thread has an assigned **500us** of execution time in total. Depending upon system configuration, assigned execution time varies linearly as follows:

SYSTEM CLOCK	*EXECUTION TIME (CYCLES)*
40 MHz	20000
20 MHz	10000
16 MHz	8000
10 MHz	5000
8 MHz	4000

To accomplish CPU workload balancing, execution time is partitioned among the different threads using a time slicing approach as described in the following table:

TIME SLICING SCHEME						
TIME (MS)	*1MS TASKS*	*2MS A TASKS*	*2MS B TASKS*	*10 MS TASKS*	*50 MS TASKS*	*100 MS TASKS*
0						
0.5	X					
1.0		X				
1.5	X					
2.0			X			
2.5	X					
3.0		X				
3.5	X					
4.0			X			
4.5	X					
5.0		X				

5.5	X					
6.0			X			
6.5	X					
7.0		X				
7.5	X					
8.0			X			
8.5	X					
9.0		X				
9.5	X					
10.0			X	X		
…	…	…	…	…	…	…
49.0		X			X	
…	…	…	…	…	…	…
99.5	X					X
…	…	…	…	…	…	…

The task scheduler uses a hardware timer interrupt to provide a **500us** time base. The microcontroller processes this periodic interrupt and set/clear the appropriate thread flags to be processed by the “SchM_Callback()” function, who is in charge of calling each of the software tasks both from low-level and application-level.

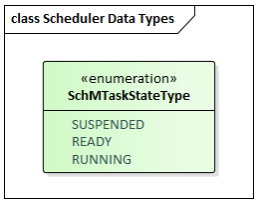
3. API SPECIFICATION

This section describes the API from the Task Scheduler module functions.

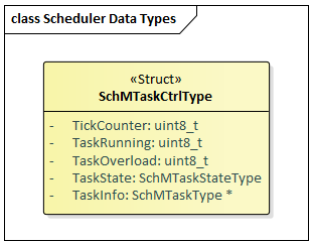
3.1. TYPE DEFINITIONS

3.1.1. PRIVATE DATA TYPES

Task state types, task transitions can be found in Scheduler State Machine chapter.

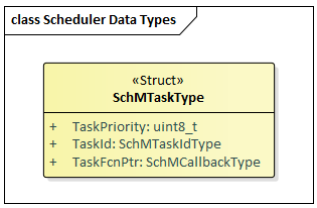


The task control type is an internal data used by the scheduler.



3.1.2. PUBLIC DATA TYPES

This data type provides support to configure the task id, task priority and task function pointer.



3.2. FUNCTION DEFINITIONS

3.2.1. SCHM_INIT

Service Name	SchM_Init	
Syntax	void SchM_Init (SchMTaskType *TaskArray)	
Sync/Async	Synchronous	
Reentrancy	Non-reentrant	
Param (in)	SchMTaskType	*TaskArray
Param (out)	None	
Return Value	None	
Description	This function performs the basic initialization of the scheduler.	

3.2.2. SCHM_START

Service Name	SchM_Start	
Syntax	void SchM_Start (void)	
Sync/Async	Synchronous	
Reentrancy	Non-reentrant	
Param (in)	None	
Param (out)	None	
Return Value	None	
Description	This function starts the Periodic Interrupt based on the SysTick Module of the microcontroller. It causes a periodic interrupt every 500us.	

3.2.3. SCHM_STOP

Service Name	SchM_Stop	
Syntax	void SchM_Stop (void)	
Sync/Async	Synchronous	
Reentrancy	Non-reentrant	
Param (in)	None	
Param (out)	None	
Return Value	None	
Description	This function stops the execution of the SysTick module of the microcontroller. It disables the timer, thus halting the time-base of the task scheduler.	

3.2.4. SCHM_SCHEDULER

Service Name	SchM_Scheduler	
Syntax	void SchM_Scheduler (void)	
Sync/Async	Synchronous	
Reentrancy	Non-reentrant	
Param (in)	None	
Param (out)	None	
Return Value	None	
Description	Multi-thread round robin task Scheduler (non pre-epmtive). It calls the different tasks based on the task state being in Ready state. Sets the task to Running state prior execution and Suspended when the execution is finished.	

3.2.5. SCHM_CALLBACK

Service Name	SchM_Callback	
Syntax	void SchM_Callback (void)	
Sync/Async	Synchronous	
Reentrancy	Non-reentrant	
Param (in)	None	
Param (out)	None	
Return Value	None	
Description	Periodic callback function invoked by the SysTick Hanlder. This interrupt is the core of the task scheduler. It is executed every 500us. It defines 3 basic threads from which other 3 threads are derived: a) 1ms thread (basic) -> 100ms thread (derived)	

	b) 2ms A thread (basic)-> 50ms thread (derived) c) 2ms B thread (basic)-> 10ms thread (derived) It partitions core execution time into time slices (500us each one). This arrangement assures core will have equal task loading across time.

3.2.6. SchM_SchedulePoint

Service Name	SchM_SchedulePoint	
Syntax	void SchM_SchedulePoint (void)	
Sync/Async	Synchronous	
Reentrancy	Non-reentrant	
Param (in)	None	
Param (out)	None	
Return Value	None	
Description	This function allows activated higher priority tasks to run. Sets the Suspended Task to Ready and the higher priority task in Running state prior its execution.	

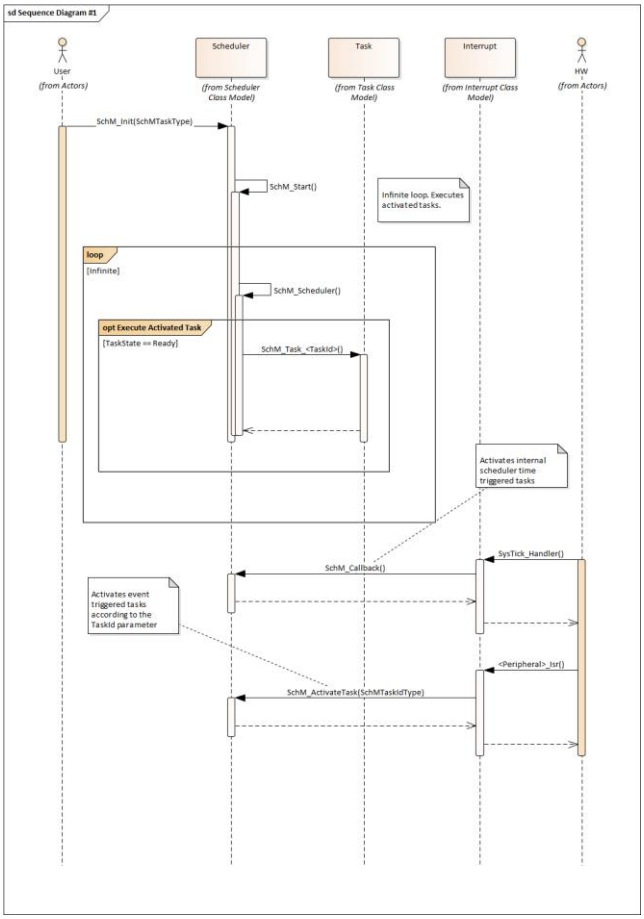
3.2.7. SchM_ActivateTask

Service Name	SchM_ActivateTask	
Syntax	void SchM_ActivateTask (SchMTaskIdType TaskId)	
Sync/Async	Synchronous	
Reentrancy	Non-reentrant	
Param (in)	SchMTaskIdType	TaskId
Param (out)	None	
Return Value	None	
Description	This function supports the tasks activation. Sets the task state to Ready.	

3.3. SEQUENCE & STATE MACHINE DIAGRAMS

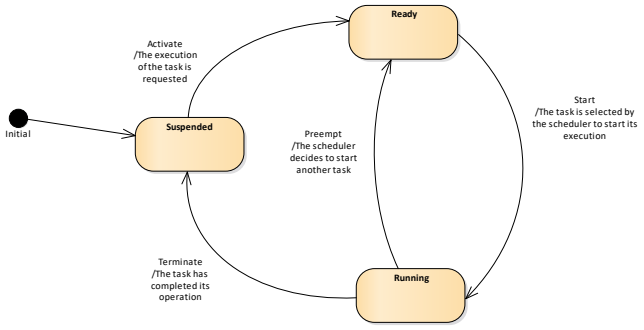
3.3.1. SCHEDULER INITIALIZATION & OPERATION

This sequence diagram depicts the general behavior of the scheduler functionality and its interaction with the external HW interrupts to activate the time triggered tasks (internally activated by the scheduler) and the event triggered tasks (external functionality of the scheduler).



3.3.2. SCHEDULER TASK STATE MACHINE

This state diagram depicts the internal transitions of a task in the scheduler.



3.3.3. SCHEDULEPOINT

This sequence diagram shows the general behavior of the Schedule Point function.